Cost-Efficient Training for Automated Algorithm Selection

Erdem Kuş[1] **Özgür Akgün**[1] **Nguyen Dang**[1] **Ian Miguel**[1]

[1]School of Computer Science, University of St Andrews, Scotland

Abstract When solving decision and optimisation problems, many competing algorithms have complementary strengths. Typically, there is no single algorithm that works well for all instances of a problem. Automated algorithm selection has been shown to work very well for choosing a suitable algorithm for a given instance. However, the cost of training can be prohibitively large due to the need of running all candidate algorithms on a set of training instances. In this work, we explore reducing this cost by selecting specific instance/algorithm combinations to train on, rather than requiring all algorithms for all instances. We approach this problem in three ways: using active learning to decide based on prediction uncertainty, augmenting the algorithm predictors with a timeout predictor, and collecting training data using a progressively increasing timeout. We evaluate combinations of these approaches on six datasets from ASLib and present the reduction in labelling cost.

1 Introduction

Solving combinatorial optimisation problems is a challenging task, where often multiple approaches compete to offer the most effective solution. Typically, different algorithms are better suited for different problem instances. Automated Algorithm Selection (AS) [20, 12, 11] techniques focus on building machine learning models to predict the most suitable algorithm for a particular problem instance. They have proven highly effective, often leading to significant improvements in computational efficiency and overall problem-solving capability (e.g., [25, 26, 1]). However, to collect (i.e., label) the training data, we need to evaluate the performance of all algorithms under study on a set of training instances. This *labelling cost* can be computational expensive, limiting the scalability and practicality of current algorithm selection methods.

Herein we propose to select instances for training AS models iteratively. This setting is called Active Learning (AL)[5], a popular cost-effective family of techniques for training machine learning models. We focus on AS scenarios that aim to optimise algorithm runtime, where each run of an algorithm on a problem instance is limited by a typically large cutoff time and so full information about performance data of timeout cases is expensive to collect but not necessarily useful.

In a typical AL scenario, the labelling cost of each data point is assumed to be uniform [5]. Our AS context is more complicated: within the cutoff, there is often significant variance in the cost of running an algorithm. Therefore, we propose two strategies within the active learning setting to reduce labelling cost, namely *timeout predictor* and *dynamic timeout*, to improve learning efficiency. We evaluate these strategies in combination with two methods for selecting new instances in the active learning setting: an uncertainty-based and a naive random selection method. Experimental results on six scenarios from ASLib [2], the standard benchmarking library for algorithm selection demonstrate the effectiveness of our proposed strategies in the AS context: in most cases, we can achieve 100% of the predictive power of an AS method that uses all of the training data, while requiring less than 20% of total labelling cost when all proposed strategies are used in combination.

2 Frugal Algorithm Selection

In this section, we first explain the underlying AS model adopted in this work. We then describe how to adopt active learning to select a subset of instances for labelling during the training (Section 2.1)

and the two additional strategies we propose, namely *timeout predictors* (Section 2.2) and *dynamic timeout* (Section 2.3), to improve the saving in labelling cost during the instance selection process.

The AS model used in our experiment follows a pairwise classification approach, as it has been shown to be effective for several AS scenarios (e.g. [26]). The approach is a collection of binary classifiers, each designed to compare a pair of algorithms to determine which one is faster for a given instance. The algorithm with the highest number of votes across all classifiers is chosen as the best option. Following previous work [26], we use a random forest for each classifier.

Passive learning – an AS model trained using the entire training set. This is the baseline for investigating the effectiveness of our frugal AS methods: we want the frugal AS to achieve the same performance as this passive learning model, while using a significantly less amount of training data.

Frugal methods – We explore three configuration options, each offering two alternatives, to create a range of strategies for frugal algorithm selection. The first configuration option is instance selection, which involves comparing uncertainty-based selection (focusing on potentially informative instances) with random selection. The second configuration option is whether we use timeout predictors and the third configuration option is whether we use dynamic timeouts. The pseudocode for these configuration options can be found in Appendix A.

2.1 Instance Selection: Uncertainty-based vs Random

In our frugal algorithm selection methods, we begin by training all classifiers on a small number of randomly selected instances. The remaining instances in the training set are kept in a pool of candidates. For each classifier we maintain a separate pool of candidates: this allows us to run an instance on a subset of the algorithms instead of necessarily running it on all algorithm options. This flexibility can be particularly useful when some algorithms tend to timeout very often and hence take up a lot of resources unnecessarily.

At each training iteration, we select N pairs of algorithm-instance combinations from the available unlabelled set of instances. The selection can be done *randomly*, or alternatively, based on a common selection strategy from the active learning literature where the *uncertainty of the prediction* [5] is taken into account: we prioritise selecting the instances where the classifier returns the highest uncertainty in its prediction. In our approach, we put all candidate pairs of classifiers and instances in a single table, sorted by their uncertainty. This would enable the classifiers with a high level of uncertainty to query more instances in comparison to those with very low levels of uncertainty.

2.2 Timeout Predictor

When one algorithm solves an instance quickly and another times out, we gain little additional information by allowing the slower algorithm to run to completion. We take into account this fact in our frugal AS approach via introducing additional binary classifiers whose task is to predict whether an unseen instance will time out for a specific algorithm. The hypothesis is that training a timeout predictor is a simpler learning task and therefore can be trained with less data. We adjust our voting mechanism by excluding pairs involving an algorithm predicted to timeout. If all algorithms are predicted to timeout, we retain all options and use the full set of pairwise predictors.

2.3 Dynamic Timeout

The dynamic timeout strategy begins with an initially defined timeout period and incrementally increases it up to the original cutoff time. After the initial training phase in active learning, the algorithms selected for querying are executed on the selected data within the current time limit. An algorithm that fails to solve the example within this specified time is classified as a timeout for the active time limit. This approach is intended to minimise labelling costs by initially running instances with a short time limit. Hence, resources are not wasted on instances that both algorithms

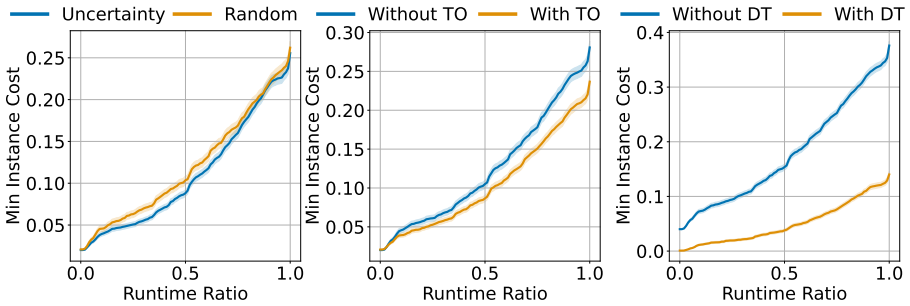

Figure 1: Results aggregated by configuration option. Instance selection (random vs uncertainty-based) does not make a big difference, timeout predictor (TO) improves runtime ratio slightly, dynamic timeout (DT) improves runtime ratio significantly.

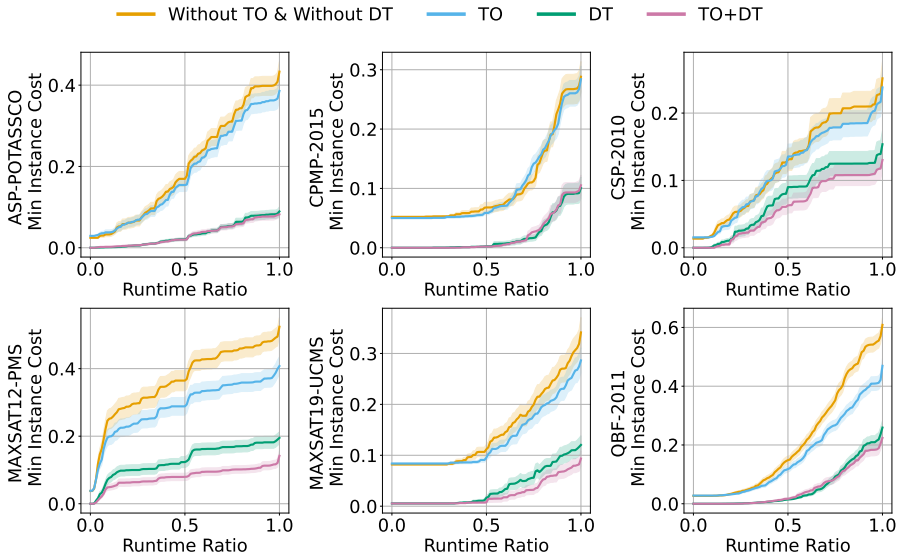

Figure 2: Performance of different timeout configurations (aggregated by instance selection). Notably, the combination of timeout predictor and dynamic timeouts (TO+DT) and the standalone dynamic timeout option exhibit significantly better performance.

are likely to fail to solve in the early stages. In single timeout cases, where one of the two algorithms can solve the instance, samples can be labelled at a lower cost. The condition for increasing the timeout is based on the model's performance on a validation set, derived from the initial dataset split. If the model predictions reach a plateau on the validation set, we increase the timeout at a specific rate. Note that when timeout predictors and dynamic timeouts are used in combination, we train the timeout predictors with respect to the particular timeout value at a given moment.

3 Experimental Results

We evaluate the performance of all eight combinations of the three frugal AS strategies described in the previous section using six datasets from ASLib, chosen for their wide-ranging characteristics. We refer to Appendix E for detailed descriptive statistics of the selected datasets. Our experiments adopt 10-fold cross-validation per dataset. The evaluation of each configuration on each fold is also repeated 5 times. We refer to Appendix D for details of our experiment setup and the hyper-parameter values chosen.

Our results, shown in Figure 1 and Figure 2, demonstrate the efficiency of the frugal AS methods in terms of data utilization. The horizontal axis represents the ratio between the performance of the frugal method and passive learning, with a range of $[0, 1]$, where 1 indicates equivalent performance. The vertical axis, capped at 1, shows the percentage of labeling cost required to achieve this performance. Notably, the cost shown is the lowest among all splits and seeds, emphasizing cost-effectiveness. Our approach remains robust, as adding more data does not harm prediction performance. Full details, including the source code and data, are available at `https://github.com/stacs-cp/CP2024-Frugal`.

Key findings

Even in the worst configuration of frugal algorithm selection, we are able to reduce the labelling cost without sacrificing predictive performance in comparison to passive learning (See Figure 2). In several cases the training effort is reduced to 10% of the labelling cost of passive learning. It is clear that our frugal approaches are able to reduce the training cost independent of configuration.

Figure 1 presents an overview of the entire set of experiments. The results are aggregated one configuration option at a time, combining results of all configurations that share, for example, Uncertainty as the instance selection method in the first plot. Overall, instance selection strategy does not make a big difference, using timeout predictors (TO) improves performance slightly, and using dynamic timeouts (DT) improves performance significantly.

The observation that uncertainty-based instance selection is not better than random may not be unexpected, as selecting instances purely based on informativeness (via prediction uncertainty) does not take the cost of running an instance on a particular algorithm into account. In settings where there is a uniform cost across all candidates, uncertainty-based selection may perform better than random. In our setting, however, some samples are significantly cheaper than others. In this setting, we want to balance *'bang for the buck'*: maximise how much information is gained per time spent. Therefore having explicit timeout predictors and a dynamic timeout strategy makes a more significant contribution.

Since instance selection strategy does not make a big difference and we observe an interesting interaction between TO and DT, we aggregate over the instance selection strategy and plot 4 options in Figure 2. We see that the combination of TO and DT provide further improvement over using each option alone, confirming the effectiveness and complementary of each strategy in runtime-based AS context.

4 Related Work

The question of how to effectively select a representative subset of benchmark instances from a large pool for a reliable and cost-effective comparison of algorithms has been investigated across different domains, including SAT [9, 16, 7], CP [17], combinatorial optimisation [18], evolutionary computation [4], and machine learning [19]. While a majority of previous work focuses on selecting instances in a static setting (i.e., all instances are chosen at once), some recent work has proposed selecting instances in an iterative fashion: Matricon et al. [17] present a statistical-based method to incrementally select instances for comparing performance between two solvers, while Fuchs et al. [7] propose an active learning-based approach for cost-effective benchmark instance selection. The key difference between those works and ours is that they focus on identifying the algorithm with the best *overall performance* across a given problem instance distribution, while ours focuses on the algorithm selection context, where the aim is to predict the best *instance-specific* algorithm.

The closest work to ours is Volpato and Song [24], where three commonly-used active learning techniques were evaluated in an AS scenario for SAT [1]. However, one drawback of their work is that they did not consider the significantly varying labelling costs among algorithms and instances,

[1]To our binary classification models, the three techniques are identical, as detailed in Appendix B.

despite it being a common characteristic of SAT scenarios. Consequently, the effectiveness of active learning for instance selection was reported based on the percentage of labelled data being saved, rather than their real saving cost.

Although the majority of active learning techniques assume the uniform labelling cost, there exist a number of works on non-uniform labelling cost settings (e.g., [21, 22, 23]). A common approach in those settings is to predict the labelling cost and to strive for a balance between informativeness and the predicted cost of a new data point. We adopt a similar technique in our work, where timeout predictors are used for identifying costly (unlabelled) data points. It can be considered a "softened" version of the cost estimation approach, as predicting the precise runtime of algorithms in the domain of combinatorial optimisation is often known to be difficult: most prominent AS techniques focus on learning the ranking among algorithms instead of trying to predict their runtime directly [8]. Note that a related technique to our dynamic timeout mechanism is adaptive capping [15, 3, 13, 10], a common strategy in automated algorithm configuration that sets a dynamic timeout based on performance of the current incumbent. Adaptive capping is not applicable to our context as we do not have performance data of all configurations on the same subset of instances, therefore, we adopt a simpler approach where the time limit is increased linearly at each iteration.

5 Broader Impact Statement

The proposed frugal algorithm approach can potentially make algorithm selection methods more efficient and cost-effective across a wide range of application domains. This could lead to improved performance and efficiency when solving challenging combinatorial optimization problems arising in domains such as SAT. Additionally, reduced costs and lower energy consumption associated with this approach may help mitigate biases related to resource availability and contribute to more eco-friendly computational practices.

6 Limitations

The approach, while promising, has limitations that need improvement. First, it relies on the assumption that the extracted instance features have good predictive power of algorithm performance. This assumption may not always hold in reality. Second, the evaluation is currently limited to a subset of ASLib datasets. Third, the newly proposed strategies (timeout predictors and dynamic timeout) are specifically tailored towards runtime AS scenarios only, while in practice, there are important AS scenarios that are based on solution quality rather than runtime, for which the additional strategies proposed cannot be applied directly in its current form.

7 Conclusion

We have proposed and evaluated a number of approaches to *frugal* algorithm selection, an active learning approach that attempts to reduce the labelling cost by using only a subset of the training data, together with a dynamic timeout strategy that uses incomplete information about the performance of algorithms in the portfolio. Our results confirm the effectiveness of the proposed approach and our analysis offers useful insights regarding the contribution of each individual component of the approach. Interestingly, the standard data selection technique in active learning does not contribute much to the overall performance, while our proposed dynamic timeout mechanism results in significant improvement in cost saving.

In future, we plan to incorporate enhancement techniques in AS into the proposed active learning framework, including the use of a pre-solving schedule [25] and cost-sensitive pairwise classification AS models [26, 14]. Other important avenues include investigating the impact of hyper-parameter tuning in the active learning setting, and developing an early-stopping mechanism to terminate the learning process once diminishing returns are observed.

References

[1] C. Ansótegui, J. Gabas, Y. Malitsky, and M. Sellmann. Maxsat by improved instance-specific algorithm configuration. *Artificial Intelligence*, 235:26–39, 2016.

[2] B. Bischl, P. Kerschke, L. Kotthoff, M. Lindauer, Y. Malitsky, A. Fréchette, H. Hoos, F. Hutter, K. Leyton-Brown, K. Tierney, and J. Vanschoren. Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, 2016.

[3] L. P. Cáceres, M. López-Ibáñez, H. Hoos, and T. Stützle. An experimental study of adaptive capping in irace. In R. Battiti, D. E. Kvasov, and Y. D. Sergeyev, editors, *Learning and Intelligent Optimization*, pages 235–250, Cham, 2017. Springer International Publishing.

[4] G. Cenikj, R. D. Lang, A. P. Engelbrecht, C. Doerr, P. Korošec, and T. Eftimov. Selector: selecting a representative benchmark suite for reproducible statistical comparison. In *Proceedings of The Genetic and Evolutionary Computation Conference*, pages 620–629, 2022.

[5] D. Cohn. *Active Learning*, pages 10–14. Springer US, Boston, MA, 2010.

[6] T. Danka and P. Horváth. modal: A modular active learning framework for python. *CoRR*, abs/1805.00979, 2018.

[7] T. Fuchs, J. Bach, and M. Iser. Active learning for sat solver benchmarking. In *Tools and Algorithms for the Construction and Analysis of Systems: 29th International Conference, TACAS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2022, Paris, France, April 22–27, 2023, Proceedings, Part I*, pages 407–425. Springer, 2023.

[8] J. Hanselle, A. Tornede, M. Wever, and E. Hüllermeier. Hybrid ranking and regression for algorithm selection. In *German Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 59–72. Springer, 2020.

[9] H. H. Hoos, B. Kaufmann, T. Schaub, and M. Schneider. Robust benchmark set selection for boolean constraint solvers. In *Learning and Intelligent Optimization: 7th International Conference, LION 7, Catania, Italy, January 7-11, 2013, Revised Selected Papers 7*, pages 138–152. Springer, 2013.

[10] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle. Paramils: an automatic algorithm configuration framework. *J. Artif. Int. Res.*, 36(1):267–306, sep 2009.

[11] P. Kerschke, H. H. Hoos, F. Neumann, and H. Trautmann. Automated algorithm selection: Survey and perspectives. *Evolutionary computation*, 27(1):3–45, 2019.

[12] L. Kotthoff. Algorithm selection for combinatorial search problems: A survey. *Data mining and constraint programming: Foundations of a cross-disciplinary approach*, pages 149–190, 2016.

[13] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, and F. Hutter. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022.

[14] M. Lindauer, H. H. Hoos, F. Hutter, and T. Schaub. Autofolio: An automatically configured algorithm selector. *Journal of Artificial Intelligence Research*, 53:745–778, 2015.

[15] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, M. Birattari, and T. Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[16] N. Manthey and S. Möhle. Better evaluations by analyzing benchmark structure. *Proc. PoS*, 2016.

[17] T. Matricon, M. Anastacio, N. Fijalkow, L. Simon, and H. H. Hoos. Statistical comparison of algorithm performance through instance selection. In *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[18] M. Mısır. Benchmark set reduction for cheap empirical algorithmic studies. In *2021 IEEE Congress on Evolutionary Computation (CEC)*, pages 871–877. IEEE, 2021.

[19] J. L. J. Pereira, K. Smith-Miles, M. A. Muñoz, and A. C. Lorena. Optimal selection of benchmarking datasets for unbiased machine learning algorithm evaluation. *Data Mining and Knowledge Discovery*, 38(2):461–500, 2024.

[20] J. R. Rice. The algorithm selection problem. In *Advances in computers*, volume 15, pages 65–118. Elsevier, 1976.

[21] B. Settles. Active learning literature survey. 2009.

[22] K. Tomanek and U. Hahn. A comparison of models for cost-sensitive active learning. In *Coling 2010: Posters*, pages 1247–1255, 2010.

[23] Y.-L. Tsou and H.-T. Lin. Annotation cost-sensitive active learning by tree sampling. *Machine Learning*, 108(5):785–807, 2019.

[24] R. Volpato and G. Song. Active learning to optimise time-expensive algorithm selection. *arXiv preprint arXiv:1909.03261*, 2019.

[25] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown. Satzilla: portfolio-based algorithm selection for sat. *Journal of artificial intelligence research*, 32:565–606, 2008.

[26] L. Xu, F. Hutter, J. Shen, H. H. Hoos, and K. Leyton-Brown. Satzilla2012: Improved algorithm selection based on cost-sensitive classification models. *Proceedings of SAT Challenge*, pages 57–58, 2012.

A Frugal Algorithm Selection Mechanism

Algorithm 1 Voting Mechanism for Algorithm Selection

Require: Set of algorithm predictors $\{P_{A_1,A_2}, P_{A_1,A_3}, \ldots, P_{A_{N-1},A_N}\}$, set of timeout predictors $\{P_{A_1,A_{1TO}}, P_{A_2,A_{2TO}}, \ldots, P_{A_N,A_{NTO}}\}$, validation set V
Ensure: Runtimes of the selected algorithms on selected set

Initialize a list to store runtimes: runtimes ← []
for each instance x in selected set S **do**
    Initialize a dictionary to count votes for each algorithm: vote_count ← {}
    Initialize a list to store selected algorithm predictors for voting: voting_predictors ← []
    if Timeout predictor is used **then**
        for each pairwise predictor P_{A_i,A_j} in algorithm predictors **do**
            if P_{A_i,A_iTO} and P_{A_j,A_jTO} predict no timeout for instance x **then**
                Add P_{A_i,A_j} to voting_predictors
            end if
        end for
    else
        voting_predictors ← all algorithm predictors
    end if
    for each predictor P_{A_i,A_j} in voting_predictors **do**
        Predict the better algorithm A_{ij} for instance x using P_{A_i,A_j}
        if A_{ij} is not in vote_count **then**
            Initialize vote_count$[A_{ij}]$ ← 0
        end if
        Increment vote_count$[A_{ij}]$ by 1
    end for
    Find the algorithm y with the maximum votes in vote_count
    Run the selected algorithm y on instance x
    Append runtime to runtimes
end for
return Sum of runtimes

Algorithm 2 Frugal Algorithm Selection

Require: Dynamic timeout flag, Timeout predictor flag, maximum timeout (3600 seconds), query size (1%), initial dynamic timeout, timeout increase rate

Ensure: Trained algorithm and timeout predictors, runtimes on validation set

Initialize the unlabeled data pool
if Dynamic timeout is used **then**
    Select N pairs of algorithm-instance combinations from the unlabeled set of instances within *initial dynamic timeout*
else
    Select N pairs of algorithm-instance combinations from the unlabeled set of instances within *maximum timeout*
end if
Train algorithm predictors $P_{A_1,A_2}, P_{A_1,A_3}, \ldots, P_{A_{N-1},A_N}$ using labeled data
if Timeout predictor is used **then**
    Train timeout predictors $P_{A_1,A_{1TO}}, P_{A_2,A_{2TO}}, \ldots, P_{A_N,A_{NTO}}$ using labeled data
end if
while There is data left for query **do**
    Create a table of uncertainty information for algorithm-instance combinations
    Sort the table by uncertainty in *descending* order
    Select the top *query size* instances
    if Timeout predictor is used **then**
        Eliminate algorithm-instance combinations that are likely to timeout using timeout predictors
    end if
    if Dynamic timeout is used **then**
        Label algorithm-instance combinations within the *initial dynamic timeout*
    else
        Label algorithm-instance combinations within the *maximum timeout*
    end if
    Delete algorithm-instance combinations from the pool if they do not timeout
    Train algorithm predictors $P_{A_1,A_2}, P_{A_1,A_3}, \ldots, P_{A_{N-1},A_N}$ using labeled data
    if Timeout predictor is used **then**
        Train timeout predictors $P_{A_1,A_{1TO}}, P_{A_2,A_{2TO}}, \ldots, P_{A_N,A_{NTO}}$ using labeled data
    end if
    if Timeout predictor is used **then**
        validation runtime ← Call **Voting Mechanism** with validation set V, algorithm predictors, and timeout predictors
        test runtime ← Call **Voting Mechanism** with test set T, algorithm predictors, and timeout predictors
    else
        validation runtime ← Call **Voting Mechanism** with validation set V and algorithm predictors
        test runtime ← Call **Voting Mechanism** with test set T and algorithm predictors
    end if
    if Dynamic timeout is used **then**
        if validation runtime >= previous validation runtime **then**
            dynamic timeout += timeout increase rate
        end if
    end if
end while

B Analysis of Uncertainty Measurement Behaviours in Active Learning for Binary Classification

There are three main approaches for uncertainty sampling in active learning. However, in a binary classification setting (which is what we use) these approaches perform identically to each other. We explain the different approaches here. Figure 3 shows the behaviour of these uncertainty sampling methods graphically.

We implement 'Least Confidence' in our code.

- *Least Confidence*: for a given input x and an output label $\hat{y}$, we can measure the posterior probability $P(\hat{y}|x;\theta)$ of observing $\hat{y}$ given x via the current model (parameterised by θ). The Least Confidence method selects data points x^* with the smallest maximum posterior probability across all labels:

$$x^* = \underset{x}{\operatorname{argmin}} \max_{\hat{y}} P(\hat{y}|x;\theta) \tag{1}$$

- *Margin-based*: this approach takes the two highest posterior probability values for each input data point x and calculates their difference. The smaller the difference, the less certain the model is about its prediction and vice versa. More formally, let $\hat{y_1}$ and $\hat{y_2}$ the output labels with the highest and second-highest posterior probabilities for a given input x, respectively, the queried points x^* are chosen as:

$$x^* = \underset{x}{\operatorname{argmin}}\, P(\hat{y_1}|x;\theta) - P(\hat{y_2}|x;\theta) \tag{2}$$

- *Entropy-based*: this approach takes into account the posterior probability values across *all* output classes. The idea is to select the data points x^* where there is a high entropy among the predicted output labels:

$$x^* = \underset{x}{\operatorname{argmax}} - \sum_i P(\hat{y}|x;\theta) \log P(\hat{y}|x;\theta) \tag{3}$$

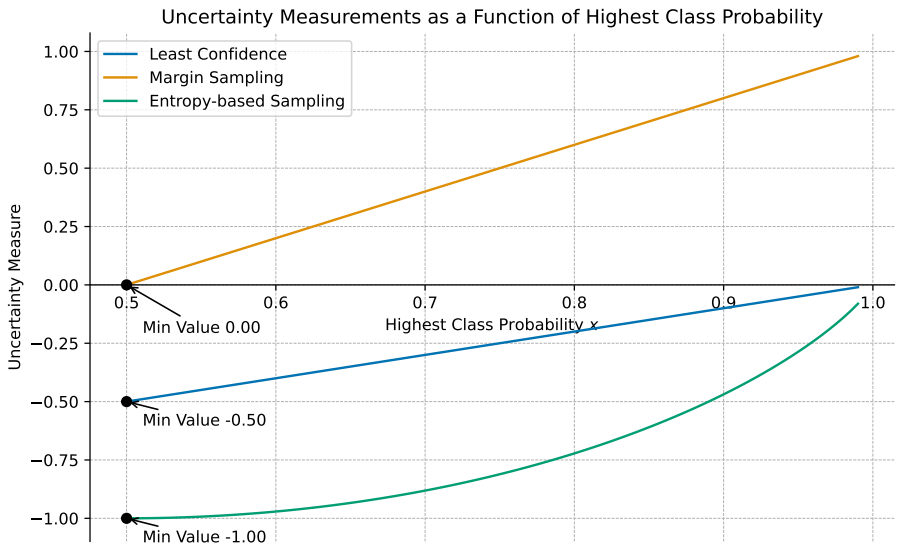

Figure 3: Uncertainty measurements as a function of the highest class probability. The red curve represents the Least Confidence uncertainty (LC) calculated as $LC = x - 1$, the green curve denotes Margin Sampling (MS) using the formula $MS = x - (1 - x)$, and the blue curve illustrates the Entropy-based method ($H(x) = -[x \log_2(x) + (1 - x) \log_2(1 - x)]$). Critical minimum values for each method are marked with black circles and annotated to emphasise the points where the uncertainty function is minimised.

C Performance of 8 Individual Configurations and Best Configuration

Figure 4 illustrates a side-by-side comparison of the following eight active learning strategies in binary classification without aggregation across configurations:

- Uncertainty Sampling (Baseline)
- Uncertainty Sampling with Timeout Predictor (TO)
- Uncertainty Sampling with Dynamic Timeout (DT)
- Uncertainty Sampling with Timeout Predictor and Dynamic Timeout (TO+DT)
- Random Sampling (Baseline)
- Random Sampling with Timeout Predictor (TO)
- Random Sampling with Dynamic Timeout (DT)
- Random Sampling with Timeout Predictor and Dynamic Timeout (TO+DT)

Figure 5 shows the performance comparison of the best configuration in the experiment, disaggregated by approach.

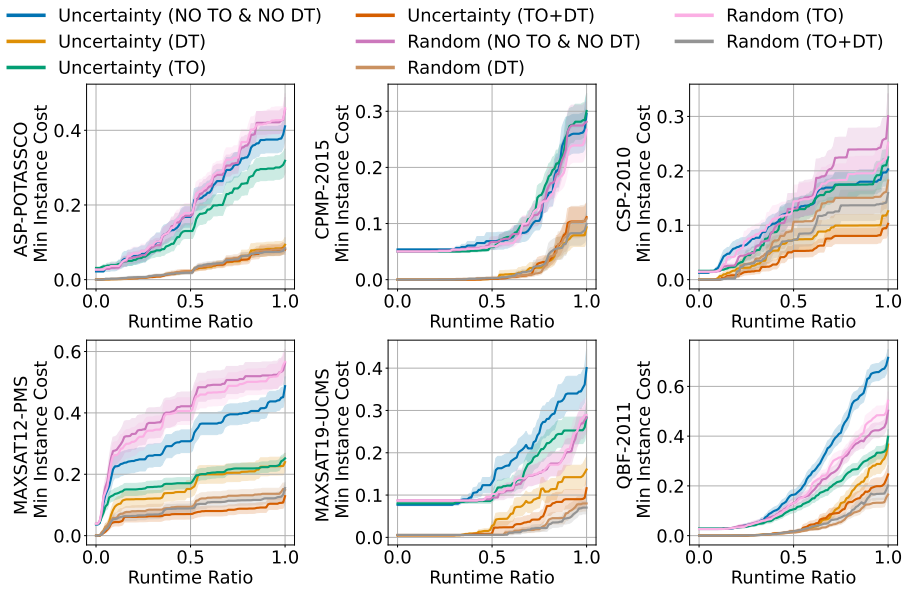

Figure 4: Comparison of performance across eight configurations as described in the paper. Each configuration was normalised according to the passive learning prediction performance ratio.

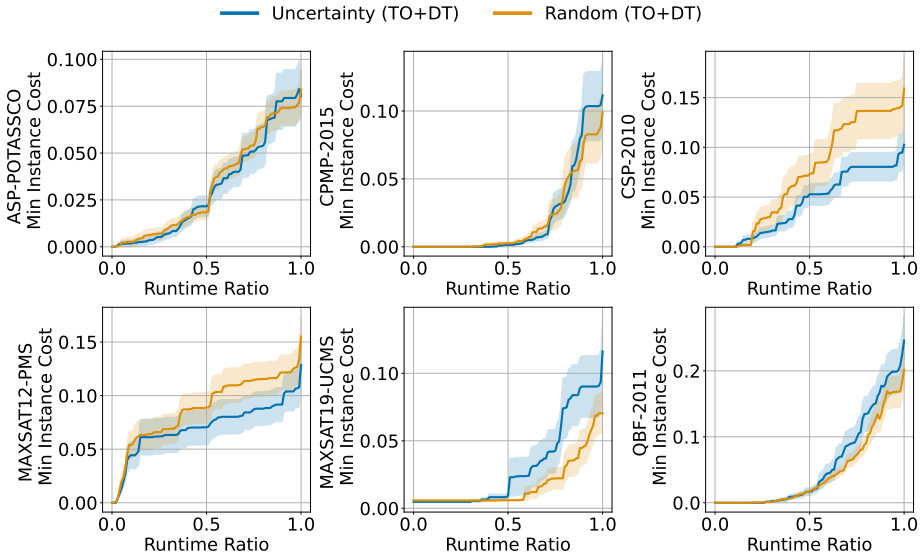

Figure 5: Comparison of TO+DT options presented in Figure 2, disaggregated by instance selection approach. No single option (uncertainty-based vs random instance selection) emerged as the clear winner.

D Experimental Setup

This study used a Random Forest classifier configured with 100 estimators and the Gini impurity measure to determine the best splits. Each tree is limited to using up to the square root of the number of features, and the depth of the decision trees is practically unlimited (with a maximum depth set to 2^{31}). Nodes require at least two samples before splitting, and bootstrapping is enabled

for sampling data when building each decision tree. These settings were determined through experimentation in the passive learning setup and were consistently used throughout the study.

We also addressed missing data by removing features where more than 20% of the instances had missing values and applied a median imputer to fill the remaining gaps.

We employed a cross-validation approach with 10 splits to validate the robustness of our study. To ensure reproducibility, we used 5 distinct seeds (7, 42, 99, 123, 12345) across our experiments, ensuring consistent generalization across multiple runs.

To determine when to increase the timeout in configurations where dynamic timeout is used, 10% of the training set was allocated as the validation set. Throughout the experiments, timeout values were scaled by a factor of 10, following the PAR10 measure.

The performance of all eight configurations of the frugal algorithm selection methods was evaluated using six datasets from ASLib, selected for their diverse characteristics. These datasets span various problem-solving domains, including one from Answer Set Programming (ASP), two from Constraint Programming (CP), two from propositional satisfiability (SAT), and one from Quantified Boolean Formula (QBF) solving. The datasets differ significantly in complexity and size, featuring between 2 to 11 algorithm options, 22 to 138 features, and 527 to 2024 instances. We chose these six specific datasets to evaluate different domains and problem characteristics comprehensively.

Additional Parameters and Configurations:

Timeout Predictor Usage: This parameter determines whether the timeout predictor is used on the system.

Timeout Limit: Sets the initial time for the dynamic timeout. We used an initial timeout of 100 seconds when employing dynamic timeout, and a fixed timeout of 3600 seconds when not using dynamic timeout.

Timeout Increase Rate: Adjusts the dynamic timeout when there is no improvement in prediction performance on the validation set. We set this rate to increase by 100 seconds when no improvement was observed.

Initial Train Size: Determines the size of the initial training set for uncertainty selection. The initial training set was created by randomly selecting 20 data points from the overall training set.

Query Size: Refers to the percentage of the dataset queried in each iteration. We set this to 1%, meaning 1% of the total pool of candidates was queried in each iteration of our experiments.

For active learning, we utilized the modAL framework [6], which facilitated the implementation of uncertainty sampling and other active learning strategies in our experiments.

This work used the Cirrus UK National Tier-2 HPC Service at EPCC (http://www.cirrus.ac.uk) funded by the University of Edinburgh and EPSRC (EP/P020267/1). The total CPU time for both the training and evaluation of the passive learning, uncertainty selection model, and random selection, was 1291.48 hours.

E Description Table of Selected Datasets

Table 1 shows key information about the datasets used in this study. It includes the time it took for the algorithms to run, the Virtual Best Solver(VBS) representing the best algorithm for each problem, and the Single Best Solver (SBS) as the best overall algorithm. While VBS is the hypothetical best, SBS serves as a benchmark for comparison against other algorithms.

Dataset	Instances	Algorithms	Features	Total Time	VBS	SBS
ASP-POTASSCO	1294	11	138	2,085h	8h	112h
CPMP-2015	527	4	22	682h	33h	134h
CSP-2010	2024	2	86	435h	49h	82h
MAXSAT12-PMS	876	6	37	1,472h	8h	85h
MAXSAT19-UCMS	572	7	54	545h	20h	52h
QBF-2011	1368	5	46	352h	28h	300h

Table 1: Descriptive statistics of selected datasets. Times rounded to the nearest whole number.

F Timeout (TO) Configuration Impact on Passive Learning

Figure 6 illustrates that the TO configuration in passive learning does not lead to a substantial improvement in performance on the test set. This observation was made in the context of selecting a baseline for comparing the performance of random and uncertainty selection.

Given that the TO configuration does not significantly enhance performance, and considering that it adds an extra layer of complexity to the model, the decision was made to use a simpler configuration for the passive learning model. Therefore, the passive learning model without a timeout predictor was chosen as the baseline for the comparisons.

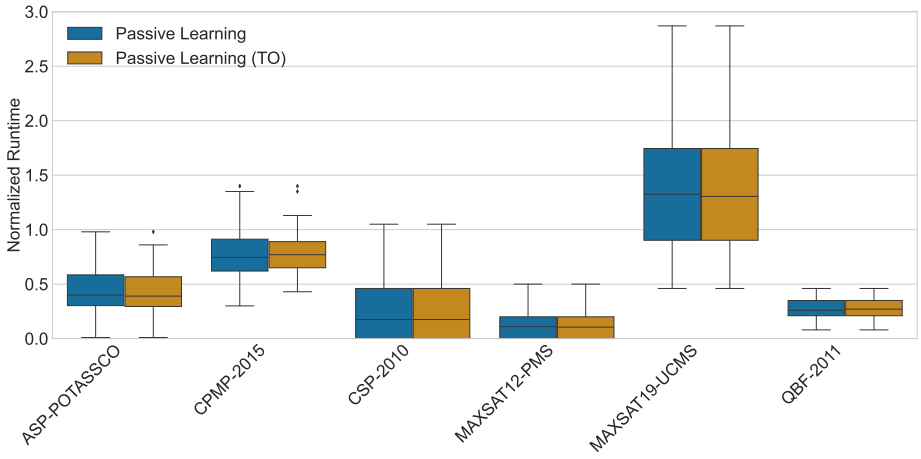

Figure 6: Comparison of Timeout (TO) Configuration Impact on Passive Learning: The graph illustrates that implementing the TO configuration in passive learning on the test set does not significantly enhance performance, yet importantly, it does not compromise prediction accuracy either.

Submission Checklist

1. For all authors...

   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]

   (b) Did you describe the limitations of your work? [Yes] See Section 6

   (c) Did you discuss any potential negative societal impacts of your work? [N/A] Our research has no identified potential negative societal impacts.

   (d) Did you read the ethics review guidelines and ensure that your paper conforms to them? `https://2022.automl.cc/ethics-accessibility/` [Yes] The paper conforms to the ethics review guidelines

2. If you ran experiments...

   (a) Did you use the same evaluation protocol for all methods being compared (e.g., same benchmarks, data (sub)sets, available resources)? [Yes]

   (b) Did you specify all the necessary details of your evaluation (e.g., data splits, pre-processing, search spaces, hyperparameter tuning)? [Yes] See Appendix D

   (c) Did you repeat your experiments (e.g., across multiple random seeds or splits) to account for the impact of randomness in your methods or data? [Yes] See Appendix D

   (d) Did you report the uncertainty of your results (e.g., the variance across random seeds or splits)? [Yes] The variance across seeds and splits are presented in all plots. The exact values can be extracted from the raw data available in the repository.

   (e) Did you report the statistical significance of your results? [No]

   (f) Did you use tabular or surrogate benchmarks for in-depth evaluations? [No]

   (g) Did you compare performance over time and describe how you selected the maximum duration? [Yes] The time dimension in our experiment is the amount of labelling cost on the training dataset. We showed how well each configuration perform when the labelling cost is increased. The maximum duration is naturally defined as the labelling cost of the full training dataset.

   (h) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)?

   (i) Did you run ablation studies to assess the impact of different components of your approach? [Yes] In our study, we conducted ablation studies by exploring eight configurations, each representing a unique combination of key components in our approach.

3. With respect to the code used to obtain your results...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results, including all requirements (e.g., `requirements.txt` with explicit versions), random seeds, an instructive `README` with installation, and execution commands (either in the supplemental material or as a URL)? [Yes] We provided an anonymous GitHub link in the paper containing all necessary code, data, and instructions to reproduce the main experimental results, including requirements.txt with explicit versions, random seeds, and an instructive README with installation and execution commands

(b) Did you include a minimal example to replicate results on a small subset of the experiments or on toy data? [No] Since we are not utilizing synthetic datasets, we don't have any small subset or toy data available for replication.

(c) Did you ensure sufficient code quality and documentation so that someone else can execute and understand your code? [Yes] We ensured code quality by providing a detailed README file that includes key parameters and an example script for running the code

(d) Did you include the raw results of running your experiments with the given code, data, and instructions? [Yes] We have included the raw experimental results in the repository along with the code, data, and instructions

(e) Did you include the code, additional data, and instructions needed to generate the figures and tables in your paper based on the raw results? [Yes] The code, additional data, and instructions required to generate the figures and tables in our paper based on the raw results are available on GitHub.

4. If you used existing assets (e.g., code, data, models)…

(a) Did you cite the creators of used assets? [Yes] We cited the creators of the dataset (Aslib), framework (modAL), and computing resource (Cirrus) used in our paper

(b) Did you discuss whether and how consent was obtained from people whose data you're using/curating if the license requires it? [Yes] We followed AsLib's rules for citing datasets, making sure we gave credit to all the sources of data in our paper.

(c) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]The data we used does not include any personally identifiable information or offensive content

5. If you created/released new assets (e.g., code, data, models)…

(a) Did you mention the license of the new assets (e.g., as part of your code submission)? [Yes] Our project uses The 3-Clause BSD License. The full license details will be available in our GitHub repository once it is made public.

(b) Did you include the new assets either in the supplemental material or as a URL (to, e.g., GitHub or Hugging Face)? [Yes] The new assets are included in the repository provided in the paper. It can be accessible through the repository URL mentioned in the paper.

6. If you used crowdsourcing or conducted research with human subjects…

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] Research does not involve any human or animal participants

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] Research does not involve any human or animal participants

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] Research does not involve any human or animal participants

7. If you included theoretical results…

(a) Did you state the full set of assumptions of all theoretical results? [N/A] We don't have any theoretical results

(b) Did you include complete proofs of all theoretical results? [N/A] We don't have any theoretical results